(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 874 306 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
28.10.1998 Bulletin 1998/44

(51) Int. Cl.6: G06F 3/12

(21) Application number: 98106731.7

(22) Date of filing: 14.04.1998

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 15.04.1997 US 842546

(71) Applicant: Xerox Corporation
Rochester, New York 14644 (US)

(72) Inventors:
- Levine, Jonathan D.
Rochester, New York 14609 (US)
- Gombert, Barry G.
Rochester, New York 14620 (US)
- Mayer, James L.
Rochester, New York 14620 (US)

(74) Representative:
Grünecker, Kinkeldey,
Stockmair & Schwanhäusser
Anwaltssozietät
Maximilianstrasse 58
80538 München (DE)

(54) Network printing system

(57) A printing system in which one or more clients communicate with a plurality of printers by way of a print server, whose architecture is characterized by a plurality of layers, is provided. In practice, one of the plurality of layers receives a request from one of the one or more clients and the request designates an operation to be performed with respect to one of the plurality of printers. The printing system includes an application layer, communicating with the one of the plurality of layers, for developing a command expression based on the request received by the one or the plurality of layers, and a communications interface including a first connectivity module and a second connectivity module, with the first connectivity module communicating with a first one of the plurality of printers and the second connectivity module communicating with a second one of the plurality of printers. The printing system further includes a routing interface, communicating with both the application layer and the communications interface, for directing the command expression to a selected one of the first connectivity module and the second connectivity module. Accordingly, the selected one of the first and second connectivity modules uses the command expression to perform at least a part of the designated operation with respect to the one or the plurality of printers.

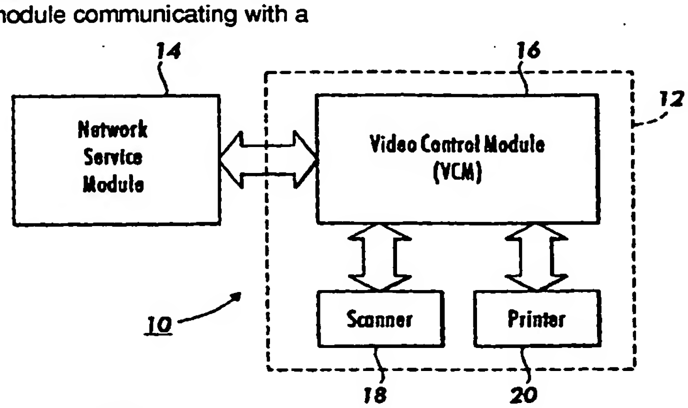


FIG. 2

EP 0 874 306 A2

Description

This invention relates generally to a printer interface for use in a network printing context and, more particularly, to an HTTP based print server adapted particularly for use with a network printing system, the HTTP based print server permitting a plurality of Internet clients to achieve Internet communication with one or more document processing devices. The architecture of the print server is such that the Internet communication can be achieved whether the one or more document processing devices has an embedded HTTP server or not.

US-A-5,220,674 discloses a local area print server which functions in cooperation with a plurality of clients and a plurality of printers to facilitate communication between the clients and the printers. The server includes various subsystems, such as a status collection subsystem that maintains a wide range of state information regarding virtually every subsystem with which the server communicates. The status collection subsystem includes a notification facility which sends reports of printing system status changes or events to appropriate network components internal and external to the local area print server that would have an interest in knowing them.

On the increasingly popular world-wide-web (www), hypertext markup language (html) specifies the display of information on a "client" computer, and hypertext transfer protocol (http) provides a neutral mechanism for the transfer of information from a "served" computer to a "client" computer over the TCP/IP network protocol. Of particular interest is the neutral aspect, in which the transfer and display of information does not depend on the client computers operating system or processor configuration, but only on the capabilities of a protocol-compliant "browser". Such software is widely available for most computers at this time. Information transferred and displayed to the client includes both static information defined in advance and dynamic information computed time. Information transferred and displayed to the client includes both static information defined in advance and dynamic information computed at the time that a client makes a request to the server. Publicly available server software often includes the common gateway interface (CGI) which allows the server to invoke a software program which may be passed user specified parameters, and whose output will be transferred to, and displayed on the client computer.

Print and document processing machines can use html and http as interfaces for control and status. These machines benefit greatly from such use for several reasons: First, development costs are lower and deployment schedules shorter since the mechanism can be used by many clients without the necessity of writing the client display software (often referred to as "user interface" or UI) for each operating system and processor that clients use. Second, it is straightforward to define multi-lingual interfaces by storing the information in multiple languages on the server, permitting the server to be accessed in multiple languages by different clients concurrently. Third, upgrades or changes can be made to the print or document processing machine's capabilities without the inconvenience of the vendor developing new client display software and of the client having to install new software on every client computer for each such upgrade.

In a network printing system, such as that disclosed by U.S. Patent No. 5,220,674, a significant number of operation requests (e.g. queries) are passed between a given client and one or more printing subsystems to determine selected information regarding a given printing subsystem, such as machine settings for or status of the given printing subsystem. These queries are best handled in an HTTP context since such context tends to be neutral across document processing platforms of varying specifications. It is believed, however, that this advantage of neutrality has not been fully exploited in the art. For instance, Hewlett-Packard ("HP") is believed to provide an arrangement in which a plurality of clients communicates with a plurality of document processing devices by way of a server. In order to use the server with the devices, however, it is necessary that the devices include a special card which is proprietary to HP. It would be desirable to provide a proxy server which is capable of performing a host of Internet operations among document processing devices some of which include embedded HTTP servers and others of which do not include embedded HTTP servers.

Additionally, in a typical proxy server an application layer communicates with a connectivity layer which permits the operation requests to be transferred between the clients and the document processing devices. Some client/server based software accommodates for the connectivity layer through an interface referred to as Middleware. In operation, Middleware includes a plurality of modules, each of which modules facilitates the transfer of operation requests between the clients and one or more target devices. For example, one module may facilitate the transfer of a query between a Microsoft based client and a Document Centre based printer ("Document Centre" is a trademark of Xerox Corp.). In one prior art scheme, a request for an operation is received from the client, via the application layer, and a set of parameters corresponding with the request is created by the application layer. In one instance, the request may seek to ascertain the values associated with selected machine settings of the target document processing device. Accordingly, the requested settings are communicated to the connectivity layer and it, in turn, obtains the values designated by the application layer.

This approach of using the application layer to set the parameter creates difficulties for the drafter of application code in that s/he may not be the one most familiar with the current parameters with which the corresponding target is associated. On the other hand, typically, those writing code for the connectivity layer are quite familiar with the current parameters with which target document processing devices are associated. Thus, it would be desirable to provide a

system in which the responsibility of obtaining values associated with a given document processing device is shifted from the application layer to the connectivity or Middleware layer.

In accordance with the disclosed invention, there is provided a printing system in which one or more clients communicate with a plurality of printers by way of a print server whose architecture is characterized by a plurality of layers. In practice, one of the plurality of layers receives a request from one of the one or more clients and the request designates an operation to be performed with respect to one of the plurality of printers. The printing system includes an application layer, communicating with the one of the plurality of layers, for developing a command expression based on the request received by the one or the plurality of layers; a communications interface including a first connectivity module and a second connectivity module, the first connectivity module communicating with a first one of the plurality of printers and the second connectivity module communicating with a second one of the plurality of printers; and a routing interface, communicating with both said application layer and said communications interface, for directing the command expression to a selected one of the first connectivity module and the second connectivity module, wherein the selected one of the first and second connectivity modules uses the command expression to perform at least a part of the designated operation with respect to the one or the plurality of printers.

Figure 1 is a perspective view of a networked digital copier suitable for responding to information requests;
Figure 2 is a block diagram depicting a multifunctional, network adaptive printing machine;
Figure 3 is a block diagram of a network controller for the printing machine of Figure 2;
Figure 4 is a block diagram showing the network controller of Figure 3 in greater detail;
Figure 5 is a block diagram in which a plurality of clients are communicatively coupled with a plurality of document processing devices (e.g. printers) by way of an HTTP proxy server;
Figure 6A is a block diagram showing a nonpreferred relationship between the routing and communications interfaces of Figure 5;
Figure 6B is a block diagram showing a preferred relationship between the routing and communications interfaces of Figure 5; and
Figure 7 is a flow diagram showing how various function calls are employed to perform operations relative to one or more document processing devices.

Referring to Figure 1 of the drawings, a digital copier system of the type suitable for use with the preferred embodiment is shown. As shown, the system includes a document feeder 1 and an operation (and display) panel 2. After desired conditions have been entered on the operation panel 2, the document feeder 1 conveys a document to a predetermined reading position on an image reading device 3 and, after the document has been read, drives it away from the reading position. The image reading device 3 illuminates the document brought to the reading position thereof. The resulting reflection from the document is transformed to a corresponding electric signal, or image signal, by a solid state imaging device, e.g., a CCD (Charge Coupled Device) image sensor. An image forming device 4 forms an image represented by the image signal on a plain paper or a thermosensitive paper by an electrophotographic, thermosensitive, heat transfer, ink jet or similar conventional system.

As a paper is fed from any one of paper cassettes 7 to the image on forming device 4, the device 4 forms an image on one side of the paper. A duplex copy unit 5 is constructed to turn over the paper carrying the image on one side thereof and again feed it to the image forming device 4. As a result, an image is formed on the other side of the paper to complete a duplex copy. The duplex copy unit 5 has customarily been designed to refeed the paper immediately or to sequentially refeed a plurality of papers stacked one upon the other, from the bottom paper to the top paper. The papers, or duplex copies, driven out of the image forming device 4 are sequentially sorted by a output device 6 in order of page or page by page.

Applications, generally 8, share the document feeder 1, operation panel 2, image reading device 3, image forming device 4, duplex unit 5, output device 6, and paper cassettes 7 which are the resources built in the copier system. As will appear, the applications include a copier application, a printer (IOT) application, a facsimile (Fax) application and other applications. Additionally, the digital copier system is coupled with a network by way of a conventional network connection 9.

Referring to Figure 2, a multifunctional, network adaptive printing system is designated by the numeral 10. The printing system 10 includes a printing machine 12 operatively coupled with a network service module 14. The printing machine 12 includes an electronic subsystem 16, referred to as a video control module (VCM), communicating with a scanner 18 and a printer 20. In one example, the VCM 16, which is described in detail in U.S. Patent No. 5,579,447 to Salgado, the disclosure of which is incorporated herein by reference, coordinates the operation of the scanner and printer in a digital copying arrangement. In a digital copying arrangement, the scanner 18 (also referred to as image input terminal (IIT)) reads an image on an original document by using a CCD full width array and converts analog video signals, as gathered, into digital signals. In turn, an image processing system (not shown), associated with the scanner 18, executes signal correction and the like, converts the corrected signals into multi-level signals (e.g. binary signals),

compresses the multi-level signals and preferably stores the same in electronic precollation (not shown).

Referring to Figure 2, the printer 20 (also referred to as image output terminal (IOT)) preferably includes a xerographic print engine. In one example, the print engine has a multi-pitch belt (not shown) which is written on with an imaging source, such as a synchronous source (e.g. laser raster output scanning device) or an asynchronous source (e.g. LED print bar). In a printing context, the multi-level image data is read out of the EPC memory, while the imaging source is turned on and off, in accordance with the image data, forming a latent image on the photoreceptor. In turn, the latent image is developed with, for example, a hybrid jumping development technique and transferred to a print media sheet. Upon fusing the resulting print, it may be inverted for duplexing or simply outputted. It will be appreciated by those skilled in the art that the printer can assume other forms besides a xerographic print engine without altering the concept upon which the disclosed embodiment is based. For example, the printing system 10 could be implemented with a thermal ink jet or ionographic printer.

Referring to Figure 3, the network service module 14 is discussed in further detail. As will be recognized by those skilled in the art, the architecture of the network service module is similar to that of a known "PC clone". More particularly, in one example, a controller 44 assumes the form of a SPARC processor, manufactured by Sun Microsystems, Inc., is coupled with a standard SBus 72. In the illustrated embodiment of Figure 3, a host memory 74, which preferably assumes the form of DRAM, and a SCSI disk drive device 76 are coupled operatively to the SBus 72. While not shown in Figure 3, a storage or I/O device could be coupled with the SBus with a suitable interface chip. As further shown in Figure 3, the SBus is coupled with a network 78 by way of an appropriate network interface 80. In one example, the network interface includes all of the hardware and software necessary to relate the hardware/software components of the controller 44 with the hardware/software components of the network 78. For instance, to interface various protocols between the network service module 14 and the network 78, the network interface could be provided with, among other software, Netware® from Novell Corp.

In one example, the network 78 includes a client, such as a workstation 82 with an emitter or driver 84. In operation, a user may generate a job including a plurality of electronic pages and a set of processing instructions. In turn, the job is converted, with the emitter, into a representation written in a page description language, such as PostScript. The job is then transmitted to the controller 44 where it is interpreted with a decomposer, such as one provided by Adobe Corporation.

Referring to Figure 4, a block diagram which further elaborates on the network controller schematic of Figure 3 is shown. In the illustrated embodiment of Figure 4, the clients 100 (each client, in Figure 3, being shown with a workstation 82 and an emitter 84) are configured with a variety of protocols, such as LPD (a protocol for UNIX), Novel network protocol, AppleTalk and DCS (a protocol for Xerox digital copiers known as the "Document Centre Systems"). Additionally, each of the clients is preferably provided with "browsing" capability which allows for communication with, for example an HTTP server for access to, among other locations, the World Wide Web. In one example, a given client employs a browser from either Netscape (2.01 or higher) or Microsoft (e.g. "nashville" (a trademark of Microsoft)). The clients communicate with the network server or electronic subsystem ("ESS") 14 by way of Connectivity Services (CS) 102. As shown in Figure 4, the ESS comprises two parts, namely a Microkernal 104 (more particularly, a partial abstract model of a Microkernal mapped into a model based on DPA ISO 10175/POSIX IEEE 1003.7) and an application specific subsystem 106.

In general, a network or point-to-point print submission originates at the Protocol Service level of the (CS) subsystem. Each Protocol Service listens on a well-known socket for a connection indication. When a Protocol Service receives the connection indication it submits a job request to the Connectivity Core. The Connectivity Core will translate this request into a DPA-compatible format and forward it to a DM subsystem 108. When the job submission is granted, the Protocol Service can submit one or more documents. Document submission is achieved by sending a document request and an I/O descriptor to the Connectivity Core. This will also be translated and forwarded to the DM subsystem.

After the document has been accepted, the Protocol Service starts to receive data from the underlying protocol stack and writes it into the I/O descriptor. This data will read on the other side of the I/O descriptor by a consumer or will be spooled somewhere in the system. When the remote client indicates that there is no more data, the I/O descriptor is closed signaling the end of this specific document. After all documents have been received, a job termination request is sent from the Protocol Service to the Connectivity Core, which then forwards it to the DM. Eventually, this request will be completed by the system, and the Protocol Service will release all resources associated with the job.

Queries normally directed from a client (Figure 4) to the printing machine 12 are, in one embodiment, processed by use of an HTTP server 107 operating in conjunction with the DM subsystem 108. In the preferred embodiment, the structure and functionality of the HTTP server resides in a proxy server 107A (Figure 5), the details of which proxy server are discussed in further detail below. It will be appreciated by those skilled in the art that while the illustrated embodiments of Figures 4 and 5 include an HTTP server, embodiments employing other server types is contemplated. That is, either server 107 or 107A could assume the form of any server that facilitates browsing on a network (as the term "browsing" is contemplated in the art) between clients and document processing devices without altering the purpose for which the preferred embodiment is intended.

The Microkernel has a Document Management (DM) subsystem that performs most of the DPA/POSIX Server functionality. The DM subsystem validates user requests, queues requests, spools document data, schedules the job for the device, and collects and maintains status information. The DM subsystem extends the DPA/POSIX Server in some aspects, since it can be configured to handle scan jobs (for filing or faxing) and copying jobs. DM provides for document sniffing, spooling, and scheduling services. Service providers, such as Document Processing 110 can register their services with DM.

Document Processing (DP), which includes the Image Frame Store (IFS) and the instantiation of at least one producer, is provided with the Microkernel. DP processes documents into images (full frame buffers or raster-scan bands, depending on the configuration of the Microkernel). The Image Frame Store assigns producers to consumers.

Essentially, the Microkernal 104 can be thought of as a generic ESS while the subsystem 106 can be thought of as an application specific ESS. That is, the Microkernal 104 contains the fundamental building blocks of a print server, while the subsystem 106 contains all of the software components necessary to, in conjunction with the Microkernal 104, provide the VCM 16 with a desired level of operability. More particularly, an Agent, which filters out all commands/requests emanating from the Microkernal 104, is designated with the numeral 114. Basically, the Agent serves as a "hook" into the generic ESS to facilitate the handling of all remote requests. The Agent works in conjunction with other services, such as Print Services 116 and Diagnostics 118, to support the operation of the VCM.

The Agent 114 also communicates with an ESS Query Utility 120 to maintain, among other things, a composite queue. While the functions of the Agent and the ESS Query Utility could be combined, they are shown as separate here in a modularized model. The ESS Query Utility also communicates with a Scan-to-File process 122, which process facilitates the filing of previously scanned documents to the network, as well as an Accounting/ Authorization/Authentication service ("AAA") 124. The AAA is used, among other things to authorize the performance of certain acts sought to be performed by a remote client. In one example, the AAA is implemented with software of the type found in Xerox' DocuSP 1.0 print server. As will appear, the AAA facilitates the preferred embodiment in that it prevents the undesirable tampering of one or more queued jobs by unauthorized system users. Additionally, as will appear, ESS Query Utility 120 can be used to obtain a host of information other than queue information. For example, the Utility 120 can be employed to obtain both machine configuration information (such as machine settings) as well as status information relating to subsystems/processes other than the queue(s).

Referring to the printing machine 12 aspect of Figure 4, a Copy Service 128 communicates with the ESS Query Utility 120 and a VCM Queue Utility 130. The Copy Service, which resides, in one example, on the controller 44 (Figure 6) performs a function, on the copy side, comparable to the DM 108. Among other things, the Copy Service, supervises the development of copy and Fax jobs as well as the management of the VCM Queue. The VCM Queue Utility communicates with and gathers queue related data from a Mark Service 132 and a suitable User Interface 134. While the functions of the User Interface and the VCM Queue Utility could be combined, they are shown as separate here in a modularized model.

The Mark Service is associated with the printer 20 (Figures 2 and 3) and a VCM Queue is associated with the User Interface, as on, for example, the Xerox Document Centre 35 digital copier. As will be appreciated by those skilled in the art, both of the Mark Service and the User Interface are key components in developing and maintaining the VCM Queue. For example, the complexion of the VCM Queue is constantly being altered as a result of activity in the Mark Service, while a significant amount of control is asserted on the VCM Queue as a function of communication with the User Interface.

Referring still to Figure 4, the HTTP server 107 further includes a common gateway interface ("CGI") designated with the numeral 140. As discussed above, the CGI, through use of suitable software, permits output responsive to user provided parameters, to be communicated to the client 100. More particularly, in practice a client user develops a query (including a set of parameters) which requests an output from a remote network system, such as the printing system 12. In one example, a request may be made with respect to information regarding the order of jobs in a queue or the current settings of the printing system. As will be appreciated by those skilled in the art, a query may be directed toward a wide range of information associated with the printing system and the remote network system being queried could include a system other than a printing system - for instance, the remote network system could include a stand-alone scanning device.

Referring to Figure 5, the proxy server, which communicatively couples the clients 100 with document processing devices 200-1 (e.g. printing system 12), 200-2, . . . 200-N, includes the following primary components:

HTTP Server 201;

Application Layer 202;

Device Data Cache & Device/User Database 204;

Routing Interface 206; and

Communications Interface 208.

In an exemplary implementation, the HTTP server 201 includes Microsoft's Internet Information Server ("Internet Information Server" ("IIS") is a trademark used by Microsoft in conjuction with web services), while the application layer 202 includes Microsoft's Internet Server applications ("Internet Server Applications" ("ISAs") is a trademark used by Microsoft in conjuction with web services), and filters (e.g. printing handlers, request for information handlers, and data update handlers). An ISA is a Dynamic Linked Library ("DLL") that gets loaded into the IIS's address space in response to a client request. The operation that takes place in response to the request happens in the ISA.

In the exemplary implementation, the device data cache & device/user database 204 (referred to below simply as "database") stores all relevant information that the proxy server has acquired from the devices 200. The stored information includes, among other things:

- a list of the devices 200,
- a current job queue for each device,
- each device's current job status,
- the current site settable for each device,
- the driver(s) for each device,
- references to appropriate strings and bitmaps to be displayed at the client for each device.

The routing interface 206, which preferably comprises a suitable application programmable interface, includes, in one example, three subsystems, namely a data supply application, a data acquisition daemon, and a device installation tool. Preferably, the data supply application (1) makes changes in site settable values in a particular device, and (2) issues job management commands to a particular device. Additionally, data acquisition daemons act as applications that initialize the communications interface 208 and gather, for instance, device, job status, and site settable information from the communications interface 208 about each registered device. In turn, the daemons populate the database with such information. Finally, the device installation tool is, as explained in further detail below, responsible for adding or deleting devices in response to client request.

The communications interface is a set of software programs intended to provide primary connectivity between the clients and the devices. In other words, a key objective of the communications interface is to provide communications between a document processing device and the proxy server's database. In practice, the communications interface acts as a liaison between the Data Acquisition Daemons and the supported devices on the customer's network. It knows how to gather and report various device status, job status, and site settable data via C++ objects and their member functions. In one instance, the communications interface includes a software module to control the communication between the database and a device type. Accordingly, a given module may be adapted to serve a single device or a whole family of devices.

As should be appreciated, while the concept underlying the disclosed server of Figure 4 can be used to implement the communications interface of Figure 5, another mode of the communications interface resides in the Document Centre System 35, a printing system provided by Xerox Corporation ("Document Centre System") is a trademark of Xerox. The following is an overview of the communications interface components, which includes information regarding middleware layer components, network abstraction layer components, infrastructure layer components and other protocols:

Middleware Layer Components

This layer implements a basic abstraction of a physical network, and provides an object representation of all supported network entities, including a network neighborhood, queues, file servers and printers and/or multi-function devices. Additionally, this layer translates the abstraction into a number of different network operating systems and protocols, including Novell, SNMP, and RPC.

Network Abstraction Layer Objects

This service, which provides an interface to a network abstraction library, is implemented as a set of C++ objects.

Infrastructure Layer Components

This layer provides low level network connectivity for one or more families of document processing platforms.

Other Protocol Support

Protocols used to facilitate operability of the above-mentioned components include, among others, TCP/IP, Netware ("Netware" is a trademark used by Novell, Inc. in conjunction with network operating systems), and LAN Manager.

The following are selected data flow scenarios contemplated for the system of Figure 5 (numbers in parentheses are keyed to this drawing):

In general:

- The database acts as a cache for all relevant device information.
- The clients receive information through/from the database.
- The database gets populated by the data supply and acquisition daemons via the communications interface, and by the device installation tool.

Client Requests for Information from server

For Device Status, Job Status, and Current Site Settable Values:

- In each case, the client requests a URL (web page) for a particular device of a particular type (1).
- The HTTP server calls the appropriate function in Application Layer (2).
- That function requests a home page (for the type of device) and its associated data (for the particular device) from the database (3) which returns the data (10).
- The application layer retrieves the appropriate web page, plugs in the data and returns the page to the client (11, 12).
- The ISA retrieves the appropriate web page, plugs in the data and returns the page to the client (11, 12).

Client Upload of Information to Proxy Server (Device / Job Management Commands)

For Altered Site Settable Values and Job Management Command:

- In each case, the client sends the data and its purpose to the HTTP server (1).
- The HTTP server calls the appropriate function in the application layer (2).
- The function of the application layer changes the appropriate cell(s) in the database (3).
- Attempting to change a value in the database triggers a data supply function call (4) that changes the values on the appropriate device via the communications interface (5, 6).
- After the data is updated on the device, the database completes the change in data.

Proxy Requests for Information from Device

For Device Status Changes, Job Status Changes, and Current Site Settable Values, Device Addition and Removal

- The data acquisition daemon registers a device with the communications interface (13). (A new device appears as a System Administrator adds a print queue on the proxy server.)
- The Middleware proceeds to acquire current device status/job status for that device (7) and return it to the data acquisition daemon (8).
- The data acquisition daemon stores the updated data in the database (9).
- When the data in the database is about to change, the database invokes functions in the application layer which sends the data updates to appropriate clients.

Device Installation

The user defines a new device by specifying an arbitrary name, a device IP address, and a device type. An LPR port and queue associated with that device is automatically set up on the HTTP server.

- The new device is then listed in the data base (16) for display at the clients.
- The new device is also registered with the communications interface (15) for access to device data.

Referring to Figures 6A and 6B, an advantageous aspect of the preferred embodiment is discussed. In a less preferred approach, as shown in Figure 6A, the application layer is responsible for making detailed requests, with respect

to, for example, obtaining device related information. That is, for those situations in which the client desires information regarding the machine settings of one of the devices 200 (Figure 5), the application, in the illustrated embodiment of Figure 6A, provides the communications interface 208 with a detailed list regarding the machine value settings that it seeks. In turn, the detailed list is communicated to a suitable module ("CIS_") in the communications interface, the module causing queries or commands of the list to be communicated to a corresponding device at which client requested values are provided. With respect to the module, each device in the disclosed system is represented as an instance of a C++ class, where each class represents a type of device. That is, one class corresponds with one type of device (e.g. a printing system), another class corresponds with another type of device (e.g. a stand-alone image capture device), and so on. As should be appreciated, this less preferred approach requires one writing code for the application layer to have relatively significant knowledge regarding the various machine settings associated with the particular device for which information is sought.

By contrast, in the preferred approach of Figure 6B, the application receives a request from a client and translates that request into a function call with an argument. In practice, as will appear from the discussion below, the information call permits the routing interface 206 to communicate to the communications interface, in a shorthand manner, what information the client desires. More particularly, the substance of the function and its argument is made available to an appropriate module in the communications interface and that module decides which values are to be provided to or which actions are to be taken with respect to a particular device. As should be appreciated, this greatly reduces the amount of commands to be provided by the application layer.

Using the underlying structure/function of the communications layer, the interface to each device looks the same to the application layer, regardless of which type of device it is and regardless of the underlying structure/function required to access the device. An exemplary list of function calls used by the proxy server is provided below:

Job Management Commands:	Hold Job(DeviceName, JobID)
	Promote Job(DeviceName, JobID)
	Cancel Job(DeviceName, JobID)
	Release Job(DeviceName, JobID)
Status Commands:	
Job Status	GetJobStatus(DeviceName)
Device Status	GetDeviceStatus(DeviceName)
Configuration Commands:	
Configuration Display	GetDeviceSettings(DeviceName)
Configuration Change	SetDeviceSettings(DeviceName, VariablesToSet)
Installation Commands:	
Device Addition	AddDevice(DeviceName, IP Address, DeviceType))
Device Deletion	DeleteDevice(DeviceName)

Referring to Figure 7, an overview of the manner in which the above calls or commands are used is provided. Initially, a request is made from a client 100 (Figure 5) to the proxy server 107A and, at step 214, the request is read at the application layer 202. Pursuant to making a function call, the application layer checks the database 204 (step 216) to determine what information about the devices exist and when that information populated the database. Preferably, the information in the database is "date stamped" so that, under certain circumstances, a client request for information can be addressed without invoking the communications interface 208 with respect to such request. For example, in one contemplated situation, relatively current machine setting information is provided from the database to the client without invoking the communications interface in direct response to such request. For ease of discussion, it will be assumed below that information or operations with respect to devices 200 are performed by invoking the communications interface with the routing interface 206 in direct response to a client's request.

At step 218, a determination as to whether the request seeks to have a device installed, i.e. added, is made. Assuming an addition is requested, a check is performed at step 220 to determine if the device to be added is already listed in the database. If the device is listed in the database, i.e. the device has already been installed, then an error

message issues (i.e. the error message is reported back to the client seeking installment), at step 222; otherwise, an Add function call is, at step 224, filled out with the name of the device to be added. In turn, the routing interface 206 (preferably an API) causes the communications interface 208 to add the new device (step 226) in accordance with the procedure described below.

Provided a request for the addition of a new device has not been made, a determination is made, at step 230, as to whether a device is to be removed. Assuming a device is to be removed or deleted, a determination, at step 232, as to whether the device is listed in the database 204 is made. If the device to be removed is not in the database, then an error message issues (i.e. the error message is reported back to the client seeking installment), at step 234; otherwise, a Remove function call is, at step 236, filled out with the name of the device to be removed. In turn, the routing interface or API 206 causes the communications interface 208 to remove the designated device (step 238) in accordance with the procedure described below.

Assuming that an installation is not required, a determination is made, at step 242, to determine whether a Get operation is requested. In order to provide a Get operation, a Get function is provided by the application layer (step 244) and its corresponding argument is filled out in accordance with the procedure below. In turn, the API 206 causes the communications interface, by way of step 246, to implement the requested Get operation. On the other hand, if a Set operation is requested (see step 248), then a Set function is provided by the application layer (step 250) and its corresponding argument is filled out in accordance with the procedure below. In turn, the API 206 causes the communications interface, by way of step 252, to implement the requested Set operation. In any event the status of an action initiated via the implementation of Figure 7 is reported back to the client requesting the action via the "STATUS" block.

The following commands/calls are implemented with selected components of the server 107A, with the components being referred to below as the "interface".

<u>Job Management Commands</u> enable a user to issue commands to jobs the interface knows about in the Window NT queue or on the destination device's hard disk if any. To the calling program, it doesn't make any difference whether the job is in a server queue or at the device. The call is the same.

<u>Status Commands</u> enable the user to get current status about a particular device or about jobs destined for a particular device:

<u>Configuration Commands</u> are used to get and set various values in the device's non-volatile memory that define the current device's configuration:

<u>Discovery or Installation Commands</u>

Numerous features of the disclosed embodiments will be appreciated by those skilled in the art:

First, a proxy server which optimizes code development is provided. More particularly, by disposing an application programming interface (API) downstream of an application layer and upstream of a communications interface, the burden of developing code at the application layer is minimized without substantially increasing the burden of code development at the communications layer. Essentially, through judicious use of the API, code development is distributed between the application layer and the communications interface in an intelligent manner.

It follows from the use of the API that requests or "calls" from the application layer to the communications layer are made exceedingly simple. Preferably, to obtain information from or perform an operation with respect to a given device, the application layer need only provide the API with a function call having a suitable argument. In turn, the API routes the function call to a subsystem in the communications interface for appropriate processing.

Second, the proxy server is usable, in an Internet context, with all sorts of document processing devices, none of which devices need be adapted specially for Internet use. Accordingly, all of the devices are usable with the server, notwithstanding their possession of a special adapter card and/or an HTTP server. Thus, the proxy server is capable of easily accommodating devices from different manufacturers with quite different designs.

Finally, the proxy server uses a database in a manner that minimizes the amount of time required to obtain device related information requested by one or more clients. In particular, time stamped information is stored periodically in the database by the API and communications interface. If the information is requested by the client and is sufficiently current when the application accesses the database, then the information is provided to the requesting client without any need to invoke the API or communications interface.

Preferably, said communications interface obtains information from a selected one of the plurality of printers, a pertinent portion of the information stored in the database is updated with the obtained information when a preselected condition is met.

Preferably, the pertinent portion of the information is characterized by an age, wherein updating of the pertinent portion occurs when the age exceeds a preselected threshold age.

Preferably, the pertinent portion of the information is updated on a periodic basis.

Claims

1. A printing system in which one or more clients communicate with a plurality of printers by way of a print server whose architecture is characterized by a plurality of layers, wherein one of the plurality of layers receives a request from one of the one or more clients and the request designates an operation to be performed with respect to one of the plurality of printers, comprising:

   an application layer, communicating with the one of the plurality of layers, for developing a command expression based on the request received by the one or the plurality of layers;
   a communications interface including a first connectivity module and a second connectivity module, the first connectivity module communicating with a first one of the plurality of printers and the second connectivity module communicating with a second one of the plurality of printers; and
   a routing interface, communicating with both said application layer and said communications interface, for directing the command expression to a selected one of the first connectivity module and the second connectivity module, wherein the selected one of the first and second connectivity modules uses the command expression to perform at least a part of the designated operation with respect to the one or the plurality of printers.

2. The printing system of claim 1, wherein said routing interface comprises an application programmable interface for interpreting the command expression to determine which of the first and second connectivity modules is to receive the command expression.

3. The printing system of claim 1, in which the one of the plurality of printers includes a queue for storing one or more print jobs with each job in the queue being characterized by state information, wherein the command expression indicates, to the selected one of the first and second connectivity modules, that selected state information regarding one or more jobs in the queue is to be retrieved.

4. The printing system of claim 1, in which the one of the plurality of printers is characterized by a set of machine settings, wherein the command expression indicates, to the selected one of the first and second connectivity modules, that selected information regarding the set of machine settings is to be retrieved.

5. The printing system of claim 1, in which the one of plurality of printers is deletable from or addable to the printing system, wherein the command expression indicates, to the selected one of the first and second connectivity modules that the one of the plurality of printers is to be added to or deleted from the printing system.

6. The printing system of claim 1, wherein the one of the plurality of layers comprises an HTTP server responsive to hypertext transfer protocol (HTTP) requests received from any one of the one or more clients.

7. The printing system of claim 6, wherein at least one, but not all, of the plurality of printers includes an HTTP server other than the HTTP server of the one of the plurality of layers.

8. The printing system of claim 1, further comprising a database, communicating with both said application layer and said routing interface, for storing information regarding each of the plurality of printers.

9. A printing system for use in an Internet or Intranet based system where one or more clients communicate with document processing devices by way of a server whose architecture is characterized by a plurality of layers, comprising:

   an HTTP server layer for receiving a request from one of the one or more clients with the request seeking a set of information relating to one of the plurality of document processing devices;
   an application layer, communicating with said HTTP server layer, for developing a command set based on the request received at said HTTP server layer;
   a set of first interfaces communicating with the plurality of document processing devices, wherein each of the first interfaces is corresponded with at least one of the plurality of document processing devices; and
   a second interface, communicating with both said application layer and each of the first interfaces of said set of first interfaces, for directing the command set to a selected one of the first interfaces, wherein the selected

one of the first interfaces uses the command set to obtain the set of information sought by the one of the one or more clients.

10. The printing system of claim 12, further comprising a database, communicating with both said application layer and said second interface, for storing information regarding each of the plurality of document processing devices.

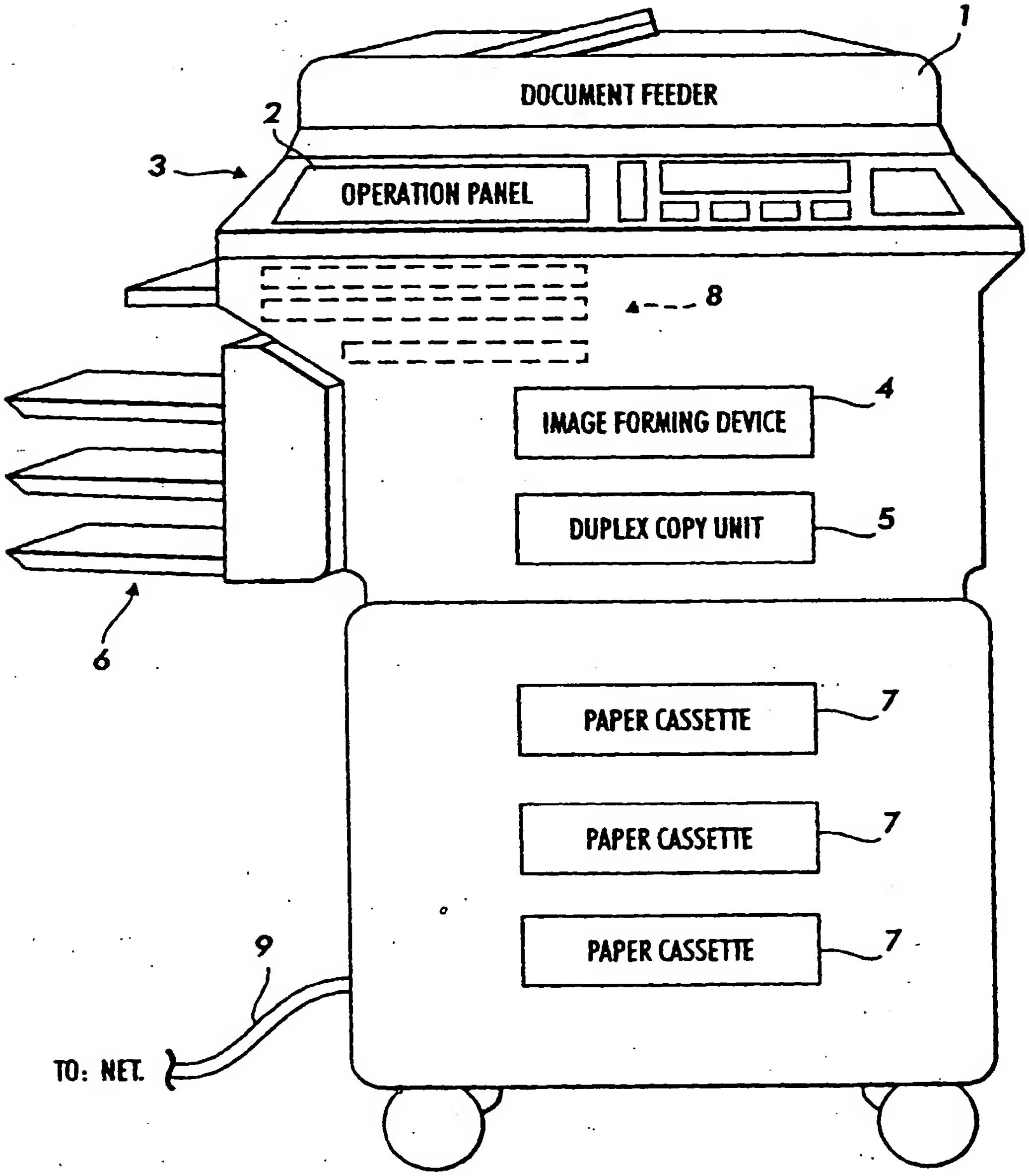


FIG. 1
PRIOR ART

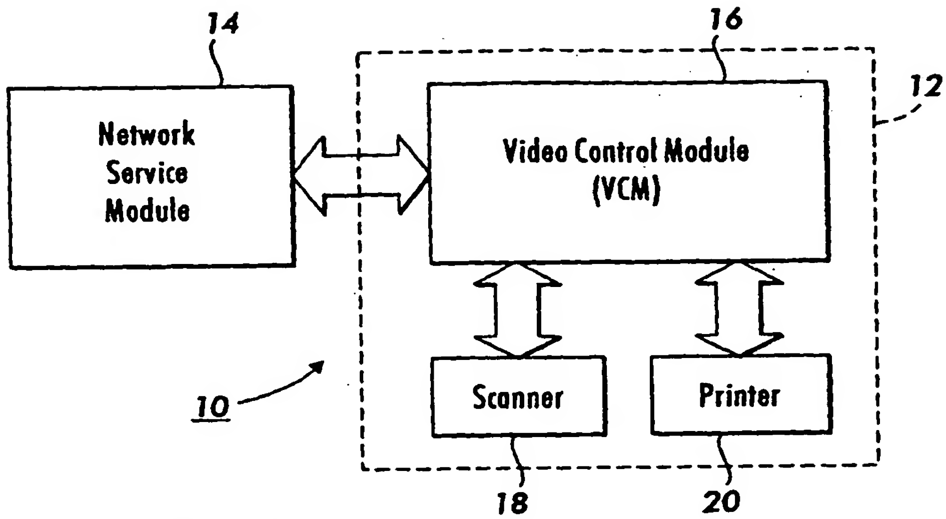

14
16
12
Network
Service
Module
Video Control Module
(VCM)
10
Scanner
Printer
18
20

FIG. 2

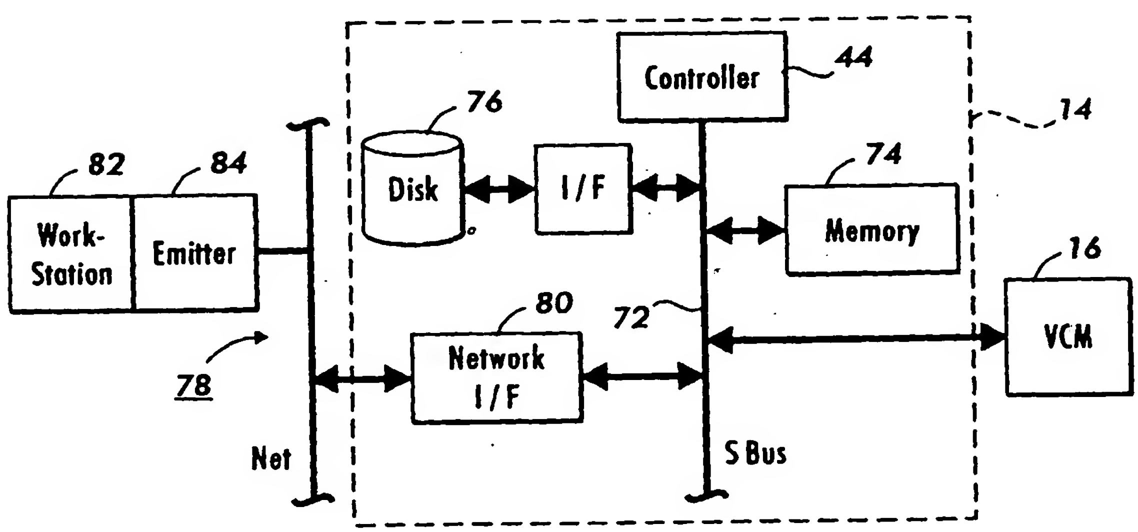

Controller
44
76
14
Disk
I/F
74
82
84
Memory
Work-
Station
Emitter
16
80
72
VCM
78
Network
I/F
Net
S Bus

FIG. 3

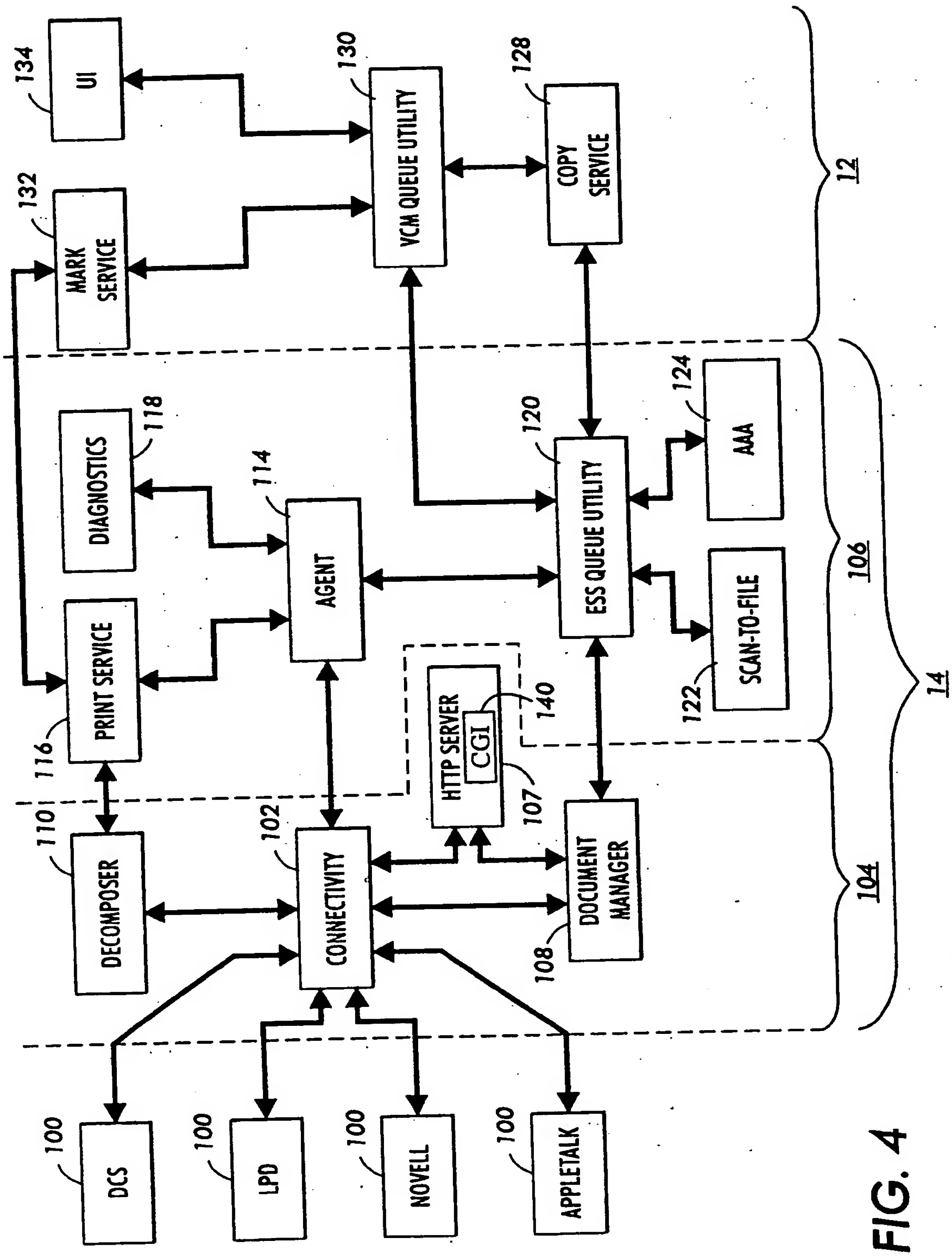

UI
134
MARK SERVICE
132
VCM QUEUE UTILITY
130
COPY SERVICE
128
12
DIAGNOSTICS
118
AGENT
114
ESS QUEUE UTILITY
120
AAA
124
SCAN-TO-FILE
122
PRINT SERVICE
116
HTTP SERVER
CGI
140
107
106
14
DECOMPOSER
110
CONNECTIVITY
102
DOCUMENT MANAGER
108
104
DCS
100
LPD
100
NOVELL
100
APPLETALK
100

FIG. 4

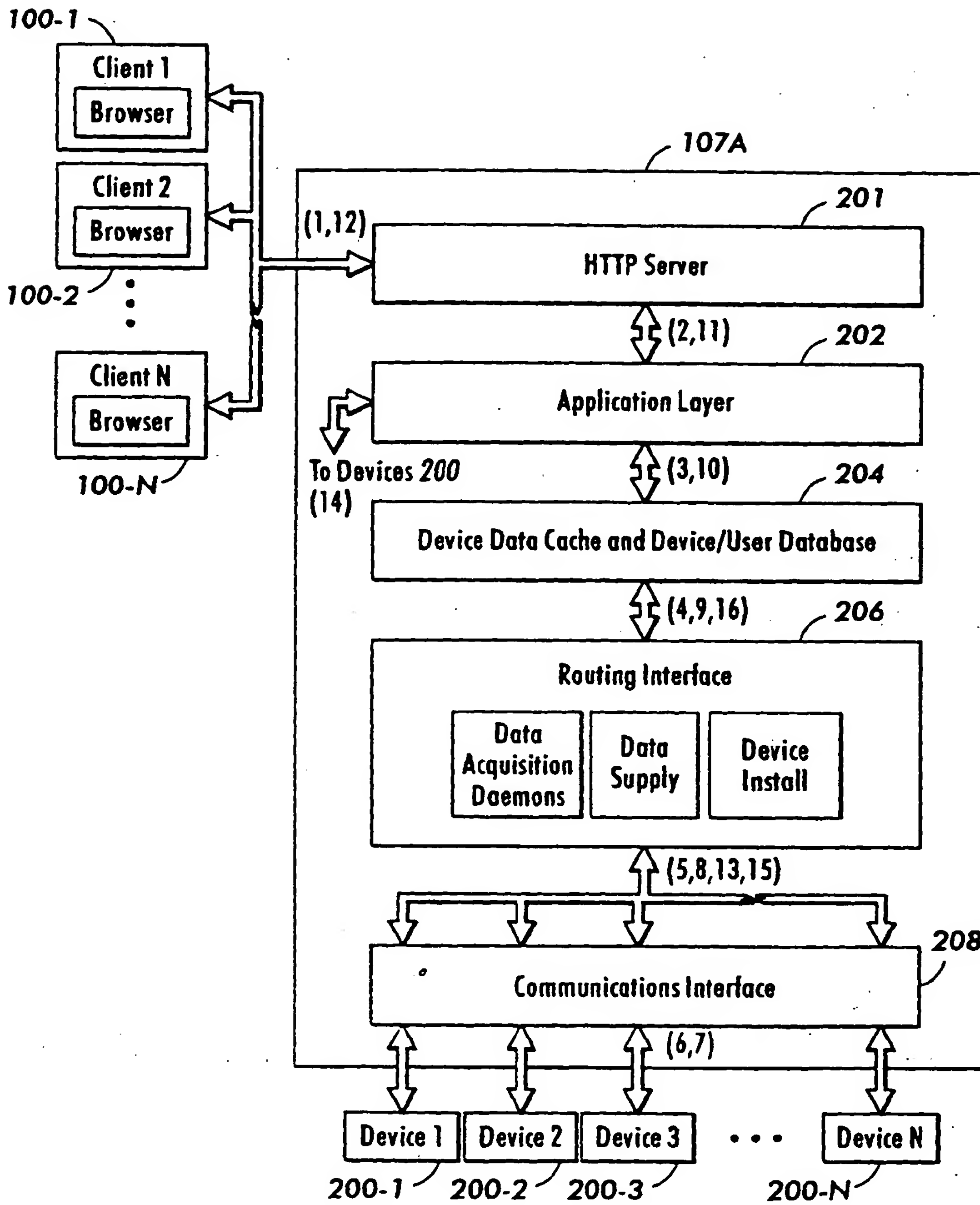

100-1
Client 1
Browser
Client 2
Browser
100-2
Client N
Browser
100-N
107A
201
(1,12)
HTTP Server
(2,11)
202
Application Layer
To Devices 200
(14)
(3,10)
204
Device Data Cache and Device/User Database
(4,9,16)
206
Routing Interface
Data Acquisition Daemons
Data Supply
Device Install
(5,8,13,15)
208
Communications Interface
(6,7)
Device 1
Device 2
Device 3
Device N
200-1
200-2
200-3
200-N

FIG. 5

FIG. 6A

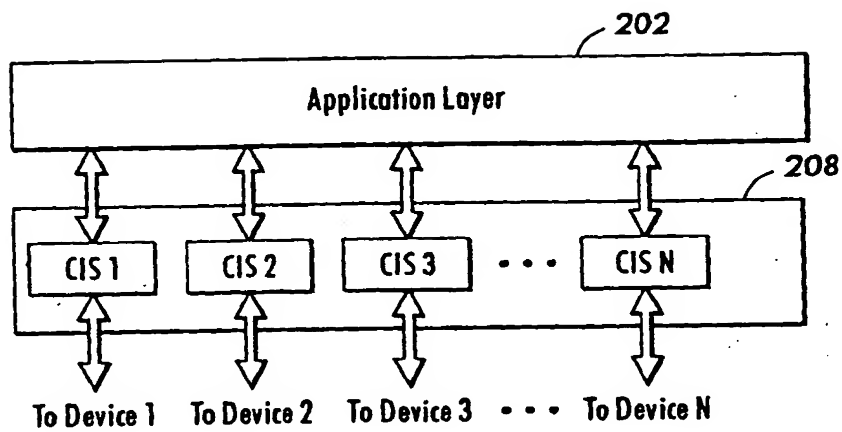
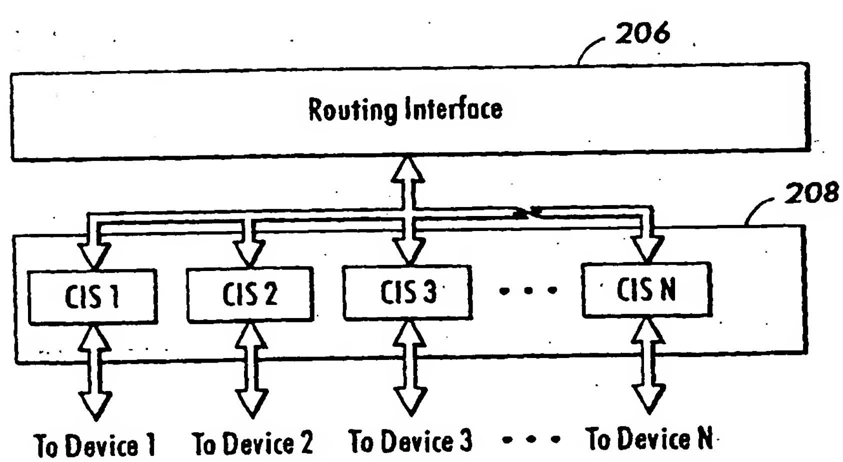

202
Application Layer
208
CIS 1
CIS 2
CIS 3
CIS N
To Device 1
To Device 2
To Device 3
To Device N

FIG. 6B


206
Routing Interface
208
CIS 1
CIS 2
CIS 3
CIS N
To Device 1
To Device 2
To Device 3
To Device N

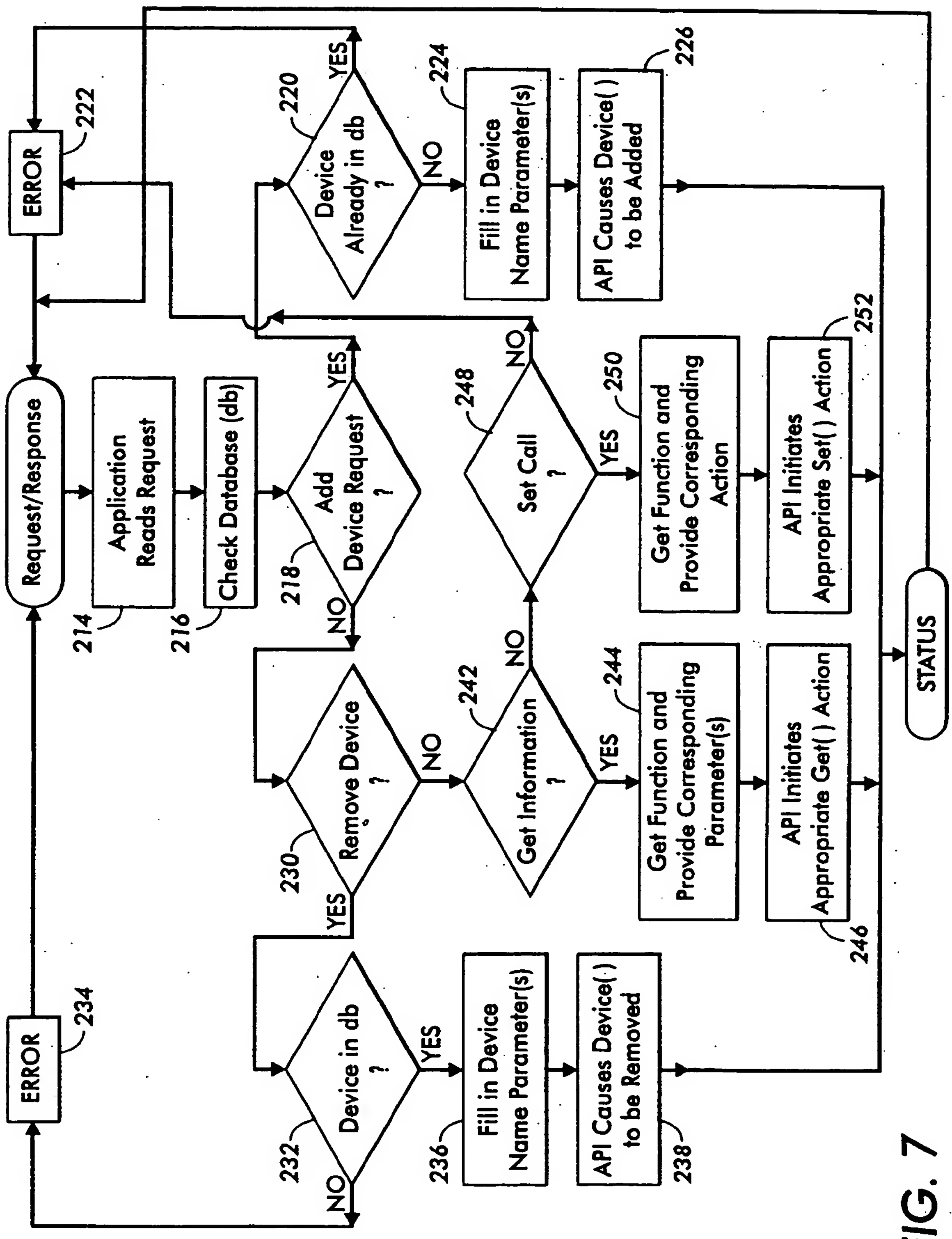

Request/Response
Application Reads Request
214
Check Database (db)
216
Add Device Request ?
218
YES
NO
Device Already in db ?
220
YES
NO
ERROR
222
Fill in Device Name Parameter(s)
224
API Causes Device( ) to be Added
226
Remove Device ?
230
YES
NO
Device in db ?
232
YES
NO
ERROR
234
Fill in Device Name Parameter(s)
236
API Causes Device( ) to be Removed
238
Get Information ?
242
YES
NO
Get Function and Provide Corresponding Parameter(s)
244
API Initiates Appropriate Get( ) Action
246
Set Call ?
248
YES
NO
Get Function and Provide Corresponding Action
250
API Initiates Appropriate Set( ) Action
252
STATUS

FIG. 7